Army Research Laboratory

Reading, Writing, and Parsing Text Files Using C++ (Updated)

by Robert J Yager

ARL-TN-0642 October 2014

NOTICES

Disclaimers

Army Research Laboratory

Aberdeen Proving Ground, MD 21005-5066

ARL-TN-0642 **October 2014**

Reading, Writing, and Parsing Text Files Using C++ (Updated)

Robert J Yager
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE							*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**							
1. REPORT DATE *(DD-MM-YYYY)* October 2014		**2. REPORT TYPE** Final					**3. DATES COVERED (From - To)** 1 January–31 July 2014
4. TITLE AND SUBTITLE Reading, Writing, and Parsing Text Files Using C++ (Updated)							**5a. CONTRACT NUMBER**
							5b. GRANT NUMBER
							5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Robert J Yager							**5d. PROJECT NUMBER** AH80
							5e. TASK NUMBER
							5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-WML-A Aberdeen Proving Ground, MD 21005-5066							**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TN-0642
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)							**10. SPONSOR/MONITOR'S ACRONYM(S)**
							11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.							
13. SUPPLEMENTARY NOTES							
14. ABSTRACT This report presents a set of functions, written in C++, that can be used to read, write, and parse text files. Updates include bug fixes and the addition of a new function.							
15. SUBJECT TERMS read, write, text file, txt, C++							
16. SECURITY CLASSIFICATION OF:				**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**		**19a. NAME OF RESPONSIBLE PERSON** Robert J Yager
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified		UU	18		**19b. TELEPHONE NUMBER** (*Include area code*) (410) 278-6689

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

Acknowledgments

I would like to thank Dr Benjamin Breech of the US Army Research Laboratory's Weapons and Materials Research Directorate. Dr Breech provided technical and editorial recommendations that improved the quality of this report.

I would also like to thank Captain Benjamin Flanders of the US Army. Captain Flanders pointed out errors in my original code and helped develop new code.

1. Introduction

This report presents a set of functions, written in C++, that can be used to read, write, and parse text files.

A summary sheet is provided at the end of this report. It presents the yIo2 namespace, which contains the ReadTextFile(), WriteTextFile(), RemoveLineComments(), RemoveBlockComments(), Parse(), Parse2D(), and PreParse() functions.

2. Background

I previously published a report titled *Reading, Writing, and Parsing Text Files Using C++*,[1] which was very similar to this report. Unfortunately, the code presented in that report contained several errors. This report fixes those errors and introduces several other improvements. A new function, PreParse(), has also been added.

The following changes from the previous code affect function behavior:

- The default value for the WriteTextFile() function's mode parameter has been changed from "w" to "wb".
- When using the default parameters, the RemoveBlockComments() function previously interpreted "*/*" as a complete block comment.
- The size of the output vector of the Parse() function was previously never less than 1, even if the function was passed a pointer that pointed to a NULL character.
- The Parse2D() function previously could only parse text that ended with a delimiter.

3. Reading and Writing Text Files

3.1 ReadTextFile() Function

The ReadTextFile() function can be used to create a character array that contains all of the information from a text file.

Note that the ReadTextFile() function uses the new[] operator to allocate memory for the character array that is pointed to by its return value. Thus, to avoid memory leaks, each use of the ReadTextFile() function should be accompanied by a use of the delete[] operator.

ReadTextFile() Code

```
inline char*ReadTextFile(//<============================READ TEXT FROM A FILE
    const char*f){//<---------------------------------THE NAME OF A TEXT FILE
  FILE*F=fopen(f,"rb");/*->*/if(!F)printf("\nCan't find \"%s\".\n",f),exit(1);
  unsigned n;/*<-*/fseek(F,0,SEEK_END),n=ftell(F),rewind(F);
  char*s=new char[n+1];/*<-*/fread(s,1,n,F),fclose(F),s[n]=0;
  return s;//.....................note that s points to newly allocated memory
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

ReadTextFile() Parameters

f — **f** points to the name of a text file.

ReadTextFile() Return Value

The ReadTextFile() function returns a pointer to the beginning of a character array that stores all of the information from the file specified by the input parameter **f**.

If the file that is specified by **f** cannot be opened, the ReadTextFile() function calls the exit() function with status code 1. Inability to open a file is typically the result of an incorrectly specified filename or path.

3.2 WriteTextFile() Function

The WriteTextFile() function can be used to write text to a file.

WriteTextFile() Code

```
inline unsigned WriteTextFile(//<========================WRITE TEXT TO A FILE
    const char*f,//<--------------------------------THE NAME OF THE TEXT FILE
    const char*s,//<-----------------------------------THE TEXT TO BE WRITTEN
    const char*m="wb"){//<---------------USE "wb" TO OVERWRITE, "ab" TO APPEND
  FILE*F=fopen(f,m);/*->*/if(!F)printf("\nCan't open \"%s\".\n",f),exit(1);
  unsigned n=fwrite(s,1,strlen(s),F);/*&*/fclose(F);
  return n;//........................number of characters written to the file
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

WriteTextFile() Parameters

f — **f** points to the name of the text file that will be written to.

s — **s** points to the text that will be written to the text file.

m — **m** specifies how text will be written to the output file. Use "wb" to overwrite an existing file. Use "ab" to append to the end of an existing file. In either case, if a file with the same name as the output filename doesn’t already exist, a new file will be created. The default value is "wb".

WriteTextFile() Return Value

The WriteTextFile() function returns the number of characters that were successfully written to the output file.

If the file that is specified by **f** cannot be opened, the WriteTextFile() function calls the exit() function with status code 1. Inability to open a file is often the result of an incorrectly specified filename or path. However, it can also be the result of a file being marked as read only, as may be the case if a file is open in another program.

3.3 ReadTextFile()/WriteTextFile() Example

The following example uses the WriteTextFile() function to create a file named "example.txt". The ReadTextFile() function is then used to read the newly created file.

```
#include "y_io_2.h"//....................................yIo2,<cstdio>{printf}
int main(){//<=====SIMPLE EXAMPLE FOR WriteTextFile() & ReadTextFile() FUNCTIONS
  yIo2::WriteTextFile("example.txt","Hello World!\n");
  char*s=yIo2::ReadTextFile("example.txt");
  printf("%s",s);//.......................display the contents of "example.txt"
  delete[]s;
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

OUTPUT:

```
Hello World!
```

---

4. Comment Removal

4.1 RemoveLineComments() Function

The RemoveLineComments() function can be used to overwrite line comments. A line comment is a comment that begins with some identifying set of characters and continues to the end of the line. Memory that is occupied by line comments is not actually freed by the RemoveLineComments() function. Instead, each line-comment character is replaced with a user-specified character.

RemoveLineComments() Code

```
  inline char*RemoveLineComments(//<=====================OVERWRITE LINE COMMENTS
      char*s,//<---------------------------TEXT THAT MIGHT CONTAIN LINE COMMENTS
      const char*c="#",//<------------------------BEGINNING-OF-COMMENT INDICATOR
      char r=' '){//<------THE CHARACTER THAT WILL BE USED TO OVERWRITE COMMENTS
    for(char*t=s;t=strstr(t,c);memset(t,r,strcspn(t,"\n\f\r")));
    return s;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

RemoveLineComments() Parameters

s — **s** points to text that will have its line comments overwritten.

c — **c** points to text that specifies the beginning of a line comment. The default value is "#".

r — **r** specifies the character that will be used to replace the characters in the line comment. The default value is a space.

RemoveLineComments() Return Value

The RemoveLineComments() function returns the input pointer **s**.

4.2 RemoveBlockComments() Function

The RemoveBlockComments() function can be used to overwrite block comments. A block comment is a comment that begins with some identifying set of characters and ends with a (potentially) different set of identifying characters. Block comments may or may not span multiple lines. Memory that is occupied by block comments is not actually freed by the RemoveBlockComments() function. Instead, each block-comment character is replaced with a user-specified character.

If the RemoveBlockComments() function encounters a beginning-of-comment indicator, but not an end-of-comment indicator, then the entire comment is ignored. The RemoveBlockComments() function does not recognize nested block comments as being nested.

RemoveBlockComments() Code

```
inline char*RemoveBlockComments(//<===================OVERWRITE BLOCK COMMENTS
    char*s,//<--------------------------TEXT THAT MIGHT CONTAIN BLOCK COMMENTS
    const char*c="/*",//<----------------------BEGINNING-OF-COMMENT INDICATOR
    const char*e="*/",//<----------------------------END-OF-COMMENT INDICATOR
    char r=' '){//<------THE CHARACTER THAT WILL BE USED TO OVERWRITE COMMENTS
  int m=strlen(c),n=strlen(e);
  for(char*t=s,*u;t=strstr(t,c);memset(t,r,u-t+n))if(!(u=strstr(t+m,e)))break;
  return s;//.........block comments without end indicators aren't overwritten
}//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

RemoveBlockComments() Parameters

s — **s** points to text that will have its block comments overwritten.

c — **c** points to text that indicates the beginning of a block comment. The default value is "/*".

e — **e** points to text that indicates the end of a block comment. The default value is "*/".

r	**r** specifies the character that will be used to replace the characters in the line comment. The default value is a space.

RemoveBlockComments() Return Value

The RemoveBlockComments() function returns the input pointer **s**.

RemoveLineComments()/RemoveBlockComments() Example

The following example code uses the RemoveLineComments() and RemoveBlockComments() functions to overwrite comments in a sample block of text.

```
#include "y_io_2.h"//....................................yIo2,<cstdio>{printf}
int main(){//<===SIMPLE EXAMPLE FOR RemoveLineComments() & RemoveBlockComments()
  char s[]="#sample line comment\nHello/*sample block comment*/World!\n";
  printf("ORIGINAL TEXT:\n%s\n\n",s);
  printf("LINE COMMENTS REMOVED:\n%s\n\n",yIo2::RemoveLineComments(s));
  printf("LINE & BLOCK COMMENTS REMOVED:\n%s\n\n",yIo2::RemoveBlockComments(s));
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

OUTPUT:

```
ORIGINAL TEXT:
#sample line comment
Hello/*sample block comment*/World!


LINE COMMENTS REMOVED:

Hello/*sample block comment*/World!


LINE COMMENTS AND BLOCK COMMENTS REMOVED:

Hello                        World!
```

5. Parsing Text

5.1 Parse() Function

The Parse() function can be used to separate the text into tokens. Tokens are blocks of text that are separated by delimiting characters. Common delimiters are commas, tabs, and spaces.

The Parse() function works by searching for tokens. When a token is found, the delimiting character that immediately follows the token is replaced by the NULL character, and a pointer to the beginning of the token is stored in a vector. Any set of consecutive delimiter characters are interpreted as a single delimiter. For example, suppose that the text "8,,,9" is parsed using the

Parse() function with **d** set to ",". The Parse() function will treat the set of 3 consecutive commas as a single delimiter. Thus, only 2 tokens will be found.

Parse() Code

```
  inline std::vector<char*>Parse(//<===================================1D PARSER
      char*s,//<----------------------------------------TEXT THAT WILL BE PARSED
      const char*d=" ,\t\n\f\r"){//<---------------------------------DELIMITERS
    std::vector<char*>V;/*<-*/for(s=strtok(s,d);s;s=strtok(0,d))V.push_back(s);
    return V;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

Parse() Parameters

s **s** points to text that will be parsed.

d **d** points to a set of token-delimiting characters. By default, spaces, commas, tabs, line feeds, form feeds, and carriage returns are all treated as delimiters.

Parse() Return Value

The Parse() function returns a vector of pointers. Each pointer points to the beginning of a token. The tokens are stored in the character array that was originally pointed to by the input parameter **s**.

Parse() Example

The following example uses the Parse() function to parse a sample block of text.

```
#include "y_io_2.h"//.............................yIo2,<cstdio>{printf},<vector>
int main(){//<===========================SIMPLE EXAMPLE FOR THE Parse() FUNCTION
  char s[]="1.0 2.0 3.0\n4.0 5.0 6.0\n7.0 8.0 9.0";
  printf("ORIGINAL TEXT:\n%s\n\n",s);
  std::vector<char*>A=yIo2::Parse(s);
  printf("TEXT PARSED USING Parse():\n");
  for(int i=0,m=A.size();i<m;++i)printf("%s%s",A[i],i==m-1?"\n":" , ");
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

OUTPUT:

```
ORIGINAL TEXT:
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0

TEXT PARSED USING Parse():
1.0 , 2.0 , 3.0 , 4.0 , 5.0 , 6.0 , 7.0 , 8.0 , 9.0
```

5.2 Parse2D() Function

The Parse2D() function can be used to separate the text contained in a character array into tokens. Tokens are separated by 2 types of delimiting characters. The first set of delimiters separates tokens within a row of data. Common examples are spaces, commas, and tabs. The

second set of delimiters separates data rows. Common examples are line feeds, form feeds, and carriage returns.

The Parse2D() function works by searching for rows, which are separated by row-delimiting characters. When a row is found, the delimiting character that immediately follows the row is replaced by the NULL character. The Parse() (not 2D) function is then used to parse the row. Any set of consecutive delimiting characters acts as a single delimiter.

Parse2D() Code

```
 inline std::vector<std::vector<char*> >Parse2D(//<=2D PARSER (CALLS 1D PARSER)
     char*s,//<----------------------------------------TEXT THAT WILL BE PARSED
     const char*d=" ,\t",//<------------------------TOKEN-DELIMITING CHARACTERS
     const char*e="\n\f\r"){//<----------------------ROW-DELIMITING CHARACTERS
   std::vector<std::vector<char*> >V;
   char*b,*c=new char[strlen(d)+strlen(e)+1];
   for(strcat(strcpy(c,d),e);*(b=s+strspn(s,c));V.push_back(Parse(b,d)))
     s=b+strcspn(b,e),!*s?s:(*s=0,++s);
   delete[]c;
   return V;//.........the number of columns per row may differ from row to row
 }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

Parse2D() Parameters

s	**s** points to text that will be parsed.
d	**d** points to a set of token-delimiting characters. Default values are spaces, commas, and tabs.
e	**e** points to a set of row-delimiting characters. Default values are line feeds, form feeds, and carriage returns.

Parse2D() Return Value

The Parse2D() function returns a vector of vectors of pointers. Each pointer points to the beginning of a token. The tokens are stored in the character array that was originally pointed to by the input parameter **s**.

Parse2D() Example

The following example uses the Parse2D() function to parse a sample block of text.

```
#include "y_io_2.h"//.............................yIo2,<cstdio>{printf},<vector>
int main(){//<========================SIMPLE EXAMPLE FOR THE Parse2D() FUNCTION
  char s[]="1.0 2.0 3.0\n4.0 5.0 6.0\n7.0 8.0 9.0";
  printf("ORIGINAL TEXT:\n%s\n\n",s);
  std::vector<std::vector<char*> >B=yIo2::Parse2D(s);
  printf("TEXT PARSED USING Parse2D():\n");
  for(int i=0,m=B.size();i<m;++i)for(int j=0,n=B[i].size();j<n;++j)
  printf("%s%s",B[i][j],j==n-1?"\n":" , ");
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

OUTPUT:

```
ORIGINAL TEXT:
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0

TEXT PARSED USING Parse2D():
1.0 , 2.0 , 3.0
4.0 , 5.0 , 6.0
7.0 , 8.0 , 9.0
```

5.3 PreParse() Function

The PreParse() function is useful for situations where tokens contain delimiting characters, such as when filenames contain spaces. The PreParse() function searches for delimiting characters that are not "protected" and replaces them with user-specified characters. Characters are assumed to be protected if they are encompassed by some sort of protecting characters (typically quotation marks, parentheses, brackets, or braces).

PreParse() Code

```
 inline char*PreParse(//<=======SELECTIVELY REPLACE DELIMITERS WITH A CHARACTER
     char*s,//<---------------------------------------------A CHARACTER ARRAY
     const char*b="\"",//<--------------BEGINNING OF IGNORED-DELIMITERS SECTION
     const char*e="\"",//<--------------------END OF IGNORED-DELIMITERS SECTION
     const char*d=" ,\t\n\f\r",//<-----------------------DELIMITING CHARACTERS
     char r='#'){//<------------------------------------REPLACEMENT CHARACTER
   const char*u;
   for(char*t=s,c=0;*t;++t){
          if(strchr(d,*t)&&!c)*t=r;
     else if(strchr(b,*t))u=b,b=e,e=u,c&1?++c:--c;
     else if(strchr(e,*t))u=b,b=e,e=u,c&1?--c:++c;}
   return s;
 }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

PreParse() Parameters

s — **s** points to the text that will have its nonprotected delimiting characters replaced.

b — **b** points to a list of characters that mark the beginning of a protected section of text. The default value is the double-quote character.

e — **e** points to a list of characters that mark the end of a protected section of text. The default value is the double-quote character.

d — **d** points to a set of delimiting characters.

r — **r** specifies the character that will be used to replace nonprotected delimiting characters.

PreParse() Return Value

The PreParse() function returns the input pointer **s**.

PreParse() Example

The following example uses the PreParse() function to prepare a sample block of text for parsing. The text is then parsed using the Parse() function.

```
#include "y_io_2.h"//.....................................yIo2,<cstdio>{printf}
int main(){//<=======================SIMPLE EXAMPLE FOR THE PreParse() FUNCTION
  char s[]="Alexis , Beatrice , Charlotte , \"Dejah Thoris\" , Eunice\n";
  printf("ORIGINAL TEXT:\n\n%s\n\n\n",s);
  printf("PRE-PARSED TEXT:\n\n%s\n\n\n",yIo2::PreParse(s,"\"","\""," ,"));
  printf("PARSED TEXT:\n\n");
  std::vector<char*>S=yIo2::Parse(s,"#\"");
  for(int i=0,n=S.size();i<n;++i)printf("%s\n",S[i]);
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

OUTPUT:

```
ORIGINAL TEXT:

Alexis , Beatrice , Charlotte , "Dejah Thoris" , Eunice



PRE-PARSED TEXT:

Alexis###Beatrice###Charlotte###"Dejah Thoris"###Eunice



PARSED TEXT:

Alexis
Beatrice
Charlotte
Dejah Thoris
Eunice
```

6. Summary

A summary sheet is provided at the end of this report. It presents the yIo2 namespace, which contains the ReadTextFile(), WriteTextFile(), RemoveLineComments(), RemoveBlockComments(), Parse(), Parse2D(), and PreParse() functions.

yIo2 Summary

y_io_2.h

```
#ifndef Y_IO_2_GUARD//       See Yager, R.J. "Reading, Writing, and Parsing Text
#define Y_IO_2_GUARD//                      Files Using C++ (Updated)" (2014)
#include <cstdio>//..fclose(),FILE,fopen(),fseek(),ftell(),printf(),SEEK_END,...
#include <cstdlib>//.....................................................exit()
#include <cstring>//...............memset(),strchr(),strlen(),strstr(),strtok()
#include <vector>//......................................................vector
namespace yIo2{//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  inline char*ReadTextFile(//<============================READ TEXT FROM A FILE
      const char*f){//<---------------------------------THE NAME OF A TEXT FILE
    FILE*F=fopen(f,"rb");/*->*/if(!F)printf("\nCan't find \"%s\".\n",f),exit(1);
    unsigned n;/*<-*/fseek(F,0,SEEK_END),n=ftell(F),rewind(F);
    char*s=new char[n+1];/*<-*/fread(s,1,n,F),fclose(F),s[n]=0;
    return s;//.....................note that s points to newly allocated memory
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
  inline unsigned WriteTextFile(//<========================WRITE TEXT TO A FILE
      const char*f,//<--------------------------------THE NAME OF THE TEXT FILE
      const char*s,//<-----------------------------------THE TEXT TO BE WRITTEN
      const char*m="wb"){//<---------------USE "wb" TO OVERWRITE, "ab" TO APPEND
    FILE*F=fopen(f,m);/*->*/if(!F)printf("\nCan't open \"%s\".\n",f),exit(1);
    unsigned n=fwrite(s,1,strlen(s),F);/*&*/fclose(F);
    return n;//.........................number of characters written to the file
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
  inline char*RemoveLineComments(//<=====================OVERWRITE LINE COMMENTS
      char*s,//<--------------------------TEXT THAT MIGHT CONTAIN LINE COMMENTS
      const char*c="#",//<-----------------------BEGINNING-OF-COMMENT INDICATOR
      char r=' '){//<------THE CHARACTER THAT WILL BE USED TO OVERWRITE COMMENTS
    for(char*t=s;t=strstr(t,c);memset(t,r,strcspn(t,"\n\f\r")));
    return s;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
  inline char*RemoveBlockComments(//<===================OVERWRITE BLOCK COMMENTS
      char*s,//<-------------------------TEXT THAT MIGHT CONTAIN BLOCK COMMENTS
      const char*c="/*",//<----------------------BEGINNING-OF-COMMENT INDICATOR
      const char*e="*/",//<----------------------------END-OF-COMMENT INDICATOR
      char r=' '){//<------THE CHARACTER THAT WILL BE USED TO OVERWRITE COMMENTS
    int m=strlen(c),n=strlen(e);
    for(char*t=s,*u;t=strstr(t,c);memset(t,r,u-t+n))if(!(u=strstr(t+m,e)))break;
    return s;//.........block comments without end indicators aren't overwritten
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
  inline std::vector<char*>Parse(//<==================================1D PARSER
      char*s,//<----------------------------------------TEXT THAT WILL BE PARSED
      const char*d=" ,\t\n\f\r"){//<---------------------------------DELIMITERS
    std::vector<char*>V;/*<-*/for(s=strtok(s,d);s;s=strtok(0,d))V.push_back(s);
    return V;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
  inline std::vector<std::vector<char*> >Parse2D(//<=2D PARSER (CALLS 1D PARSER)
      char*s,//<----------------------------------------TEXT THAT WILL BE PARSED
      const char*d=" ,\t",//<------------------------TOKEN-DELIMITING CHARACTERS
      const char*e="\n\f\r"){//<----------------------ROW-DELIMITING CHARACTERS
    std::vector<std::vector<char*> >V;
    char*b,*c=new char[strlen(d)+strlen(e)+1];
    for(strcat(strcpy(c,d),e);*(b=s+strspn(s,c));V.push_back(Parse(b,d)))
      s=b+strcspn(b,e),!*s?s:(*s=0,++s);
    delete[]c;
    return V;//.........the number of columns per row may differ from row to row
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
  inline char*PreParse(//<=======SELECTIVELY REPLACE DELIMITERS WITH A CHARACTER
      char*s,//<---------------------------------------------A CHARACTER ARRAY
      const char*b="\"",//<--------------BEGINNING OF IGNORED-DELIMITERS SECTION
      const char*e="\"",//<--------------------END OF IGNORED-DELIMITERS SECTION
      const char*d=" ,\t\n\f\r",//<------------------------DELIMITING CHARACTERS
      char r='#'){//<------------------------------------REPLACEMENT CHARACTER
    const char*u;
    for(char*t=s,c=0;*t;++t){
           if(strchr(d,*t)&&!c)*t=r;
      else if(strchr(b,*t))u=b,b=e,e=u,c&1?++c:--c;
      else if(strchr(e,*t))u=b,b=e,e=u,c&1?--c:++c;}
    return s;
  }//~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
}//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#endif
```

ReadTextFile()/WriteTextFile() Example

```
#include "y_io_2.h"//....................................yIo2,<cstdio>{printf}
int main(){//<=====SIMPLE EXAMPLE FOR WriteTextFile() & ReadTextFile() FUNCTIONS
  yIo2::WriteTextFile("example.txt","Hello World!\n");
  char*s=yIo2::ReadTextFile("example.txt");
  printf("%s",s);//........................display the contents of "example.txt"
  delete[]s;
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

OUTPUT:

```
Hello World!
```

RemoveLineComments()/RemoveBlockComments() Example

```
#include "y_io_2.h"//....................................yIo2,<cstdio>{printf}
int main(){//<===SIMPLE EXAMPLE FOR RemoveLineComments() & RemoveBlockComments()
  char s[]="#sample line comment\nHello/*sample block comment*/World!\n";
  printf("ORIGINAL TEXT:\n%s\n\n",s);
  printf("LINE COMMENTS REMOVED:\n%s\n\n",yIo2::RemoveLineComments(s));
  printf("LINE & BLOCK COMMENTS REMOVED:\n%s\n\n",yIo2::RemoveBlockComments(s));
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

OUTPUT:

```
ORIGINAL TEXT:
#sample comment
Hello/*sample comment*/ World!


LINE COMMENTS REMOVED:

Hello/*sample comment*/ World!


LINE COMMENTS AND BLOCK COMMENTS REMOVED:

Hello                    World!
```

Parse() Example

```
#include "y_io_2.h"//.............................yIo2,<cstdio>{printf},<vector>
int main(){//<==========================SIMPLE EXAMPLE FOR THE Parse() FUNCTION
  char s[]="1.0 2.0 3.0\n4.0 5.0 6.0\n7.0 8.0 9.0";
  printf("ORIGINAL TEXT:\n%s\n\n",s);
  std::vector<char*>A=yIo2::Parse(s);
  printf("TEXT PARSED USING Parse():\n");
  for(int i=0,m=A.size();i<m;++i)printf("%s%s",A[i],i==m-1?"\n":" , ");
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

OUTPUT:

```
ORIGINAL TEXT:
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0

TEXT PARSED USING Parse():
1.0 , 2.0 , 3.0 , 4.0 , 5.0 , 6.0 , 7.0 , 8.0 , 9.0
```

Parse2D() Example

```
#include "y_io_2.h"//.............................yIo2,<cstdio>{printf},<vector>
int main(){//<========================SIMPLE EXAMPLE FOR THE Parse2D() FUNCTION
  char s[]="1.0 2.0 3.0\n4.0 5.0 6.0\n7.0 8.0 9.0";
  printf("ORIGINAL TEXT:\n%s\n\n",s);
  std::vector<std::vector<char*> >B=yIo2::Parse2D(s);
  printf("TEXT PARSED USING Parse2D():\n");
  for(int i=0,m=B.size();i<m;++i)for(int j=0,n=B[i].size();j<n;++j)
  printf("%s%s",B[i][j],j==n-1?"\n":" , ");
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

OUTPUT:

```
ORIGINAL TEXT:
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0

TEXT PARSED USING Parse2D():
1.0 , 2.0 , 3.0
4.0 , 5.0 , 6.0
7.0 , 8.0 , 9.0
```

PreParse() Example

```
#include "y_io_2.h"//......................................yIo2,<cstdio>{printf}
int main(){//<=======================SIMPLE EXAMPLE FOR THE PreParse() FUNCTION
  char s[]="Alexis , Beatrice , Charlotte , \"Dejah Thoris\" , Eunice\n";
  printf("ORIGINAL TEXT:\n\n%s\n\n\n",s);
  printf("PRE-PARSED TEXT:\n\n%s\n\n\n",yIo2::PreParse(s,"\"","\""," ,"));
  printf("PARSED TEXT:\n\n");
  std::vector<char*>S=yIo2::Parse(s,"#\"");
  for(int i=0,n=S.size();i<n;++i)printf("%s\n",S[i]);
}//~~~~~~YAGENAUT@GMAIL.COM~~~~~~~~~~~~~~~~~~~~~~~~~~LAST~UPDATED~20JUL2014~~~~~~
```

OUTPUT:

```
ORIGINAL TEXT:

Alexis , Beatrice , Charlotte , "Dejah Thoris" , Eunice


PRE-PARSED TEXT:

Alexis###Beatrice###Charlotte###"Dejah Thoris"###Eunice


PARSED TEXT:

Alexis
Beatrice
Charlotte
Dejah Thoris
Eunice
```

7. References

1. Yager, RJ. Reading, writing, and parsing text files using C++. Aberdeen Proving Ground (MD): Army Research Laboratory (US); June 2013. Report No. ARL-TN-545.

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	DIR USARL RDRL WML A R YAGER